Chinese Patent No. 1368675 A

INTELLECTUAL PROPERTY BUREAU OF THE PEOPLE'S REPUBLIC OF CHINA
PUBLIC DESCRIPTION OF THE INVENTION PATENT APPLICATION
PUBLICATION NO. CN 1368675 A

Int. Cl.[7]:	G 06 F 9/30
Filing No.:	01103347.9
Filing Date:	February 1, 2001
Publication Date:	September 11, 2002

METHOD FOR BLOCKING WIN32 APPLICATION PROGRAM INTERFACE

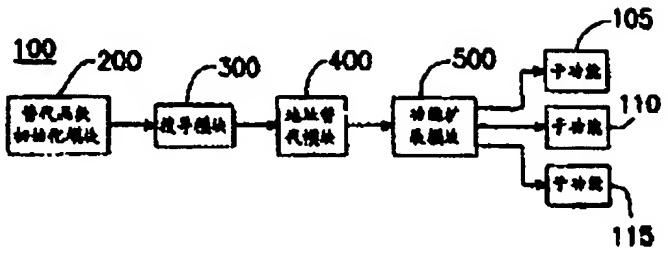
Inventors:	CHEN, Xuan Tong LIN, Guang Xin
Applicant:	Inventec Co., Ltd. Taipei, Taiwan Province
Patent Agency:	Liu Shen Attorney's Office, Beijing
Agent:	HUANG Xiao Lin

No. of pages of the claims:	2
No. of pages of the description:	5
No. of pages of the attached drawings:	4

Abstract

A method for blocking a Win32 application program interface which is comprised of a replacement function initiation step that distributes the storage space for a new definition function; a search step that is used for obtaining the jump address indicated by the JMP command in the original function to be processed; an address replacement step that starts from the jump address obtained by the search step in a search for all of the empty operation commands in the original function and omits these commands, and then the first function command in the original function is replaced with a JMP command, in which the address of the new definition function that is used to replace said original function is also entered in the jump address indicated

by the JMP command; and a function expansion step that reads out the content of the registry where new definition functions are stored.



Key: 105, 110, 115 Subfunction
200 Replacement function initiation module
300 Search module
400 Address replacement module
500 Function expansion module

Claims

1. A method for blocking a Win32 application program interface which is comprised of:

a replacement function initiation step, which is carried out step by step as follows:

(a) the storage space for a new definition function is distributed and said new definition function is analyzed

(b) when said new definition function is empty, the step is then concluded and when said new definition function is not empty, a JMP command is then entered at the initial place of said new definition function as well as an assembler code to serve as a jump address of the JMP command, in which said assembler code is an original address to be replaced by said new definition function;

(c) everything, with the exception of the JMP command and said assembler code, in said new definition function is then emptied out;

a search step, which is carried out step by step as follows:

(a) said original function to be processed is obtained;

(b) the current command of said original function is read out sequentially;

(c) when said command is empty, then the program that processes said original function using said search step is concluded;

(d) when said command is not a JMP, movement to the next command proceeds, the operation number is moved downward by one unit and then the procedure from step (b) follows;

(e) when said command is a JMP, then said jump address of said command is used as a real address for said original function to be processed and the program that processes said original function using said search step is concluded;

an address replacement step, which is carried out step by step as follows:

(a) starting from the real address of said original function, an analysis is carried out on the function command in said original function sequentially in the range of said original function;

(b) when said function command is an empty operation, then said function command is omitted while when said function command is not an empty operation, then the analysis is continued for the next said function command;

(c) the first said function command in the original function is replaced with a JMP command and the address of the new definition function that is used to replace said original function is also entered in the jump address indicated by the JMP command; and

a function expansion step, which is carried out step by step as follows:

(a) the content of the registry where the new definition functions are stored is read out;

(b) any one of said new definition functions in said registry is read out sequentially, so that said new definition function may operate at said jump address that is obtained by said address replacement step;

(c) when the contents of said registry are read out completely, then said function expansion step is concluded.

2. A method for blocking a Win32 application program interface which is comprised of:

designing a replacement function initiation step, said replacement function initiation step distributing the storage space for a new definition function and analyzing said new definition function;

designing a search step, said search step being used for obtaining the jump address indicated by the JMP command in the original function to be processed;

designing an address replacement step, said address replacement step starting from said jump address obtained by said search step in a search for all of the empty operation commands in the original function and omitting these commands, and then the first function command in the original function being replaced with a JMP command, in which the address of the new definition function that is used to replace said original function is also entered in the jump address indicated by the JMP command; and

designing a function expansion step, said function expansion step reading out the content of the registry where the new definition functions are stored and reading out sequentially any one of said new definition functions in said registry.

3. A method in accordance with Claim 2, in which said new definition function is analyzed by said replacement function initiation step and which is comprised of:

when said new definition function is empty, the step is then concluded and when said new definition function is not empty, a JMP command is then entered at the initial place of said new definition function as well as an assembler code to serve as a jump address of the JMP command, in which said assembler code is an original address to be replaced by said new definition function; and

everything, with the exception of the JMP command and said assembler code, in said new definition function is then emptied out.

4. A method in accordance with Claim 2, in which said jump address is obtained in said search step and which is also comprised of:

obtaining said original function to be processed;

reading out sequentially the current command of said original function;

when said command is empty then concluding the program that processes said original function using said search step;

when said command is not a JMP, the next command is examined; and

when said command is a JMP, then said jump address of said command is used as a real address for said original function to be processed and the program that processes said original function using said search step is concluded.

Description

The present invention pertains to a method for blocking an application program interface, in particular it pertains to a method for blocking a Win32 application program interface.

When a module is called, jumping to the address of said module occurs first, and which is then followed by a system entry. As a result, in the technology of the prior art, when an application program (AP) in a window system is in need of replacement, generally modification is carried out for the entry point of each of the application programs of each of the modules so that the original function of the system may be replaced by a function that is defined by the user. However, because a modification program is carried out for the entry point of each module of the application program in the technology of the prior art, when it is necessary to carry out modification for the entry points of application programs of a plurality of modules, each entry point of each application program of each module must be modified one by one. A method in which the modules are modified one by one not only has a slow operational speed and a small range of functions but it is also more susceptible to omission occurrence.

The objective of the present invention is to provide a method for blocking a Win32 application program interface by directly replacing the application program indicated by the Win32 application program interface so that the application program used is modified.

In order to achieve the objective of the present invention, the present invention proposes a method for blocking a Win32 application program interface which is comprised of a replacement function initiation step, which is carried out step by step as follows: (a) the storage space for a new definition function is distributed and said new definition function is analyzed; (b) when said new definition function is empty, the step is then concluded and when said new definition function is not empty, a JMP command is then entered at the initial place of said new definition function as well as an assembler code to serve as a jump address of the JMP command, in which said assembler code is an original address to be replaced by said new definition function; (c) everything, with the exception of the JMP command and said assembler code, in said new definition function is then emptied out; a search step, which is carried out step by step as follows: (a) said original function to be processed is obtained; (b) the current command of said original function is read out sequentially; (c) when said command is empty then the program that processes said original function using said search step is concluded; (d) when said command is not a JMP, movement to the next command proceeds, the operation number is moved downward by one unit and then the procedure from step (b) follows; (e) when said command is a JMP, then said jump address of said command is used as a real address for said original function to be processed and the program that processes said original function using said search step is concluded; an address replacement step, which is carried out step by step as follows: (a) starting from the real address of said original function, an analysis is carried out on the function command in said original function sequentially in the range of said original function; (b) when said function command is an empty operation, then said function command is omitted while when said function command is not an empty operation, then the analysis is continued for the next said function command; (c) the first said function command in the original function is replaced with a JMP command and the address of the new definition function that is used to replace said original function is also entered in the jump address indicated by the JMP command; and a function expansion step, which is carried out step by step as follows: (a) the content of the registry where the new definition functions are stored is read out; (b) any one of said new definition functions in said registry is read out sequentially, so that said new definition function may operate at said jump address that is obtained by said address replacement step; (c) when the contents of said registry are read out completely, then said function expansion step is concluded.

The present invention also proposes another method for blocking a Win32 application program interface which is comprised of: designing a replacement function initiation step, said replacement function initiation step distributing the storage space for a new definition function and analyzing said new definition function; designing a search step, said search step being used for obtaining the jump address indicated by the JMP command in the original function to be processed; designing an address replacement step, said address replacement step starting from

said jump address obtained by said search step in a search for all of the empty operation commands in the original function and omitting these commands, and then the first function command in the original function being replaced with a JMP command, in which the address of the new definition function that is used to replace said original function is also entered in the jump address indicated by the JMP command; and designing a function expansion step, said function expansion step reading out the content of the registry where the new definition functions are stored and reading out sequentially any one of said new definition functions in said registry.

The method for blocking a Win32 application program interface proposed by the present invention designs a replacement function initiation module, a search module, an address replacement module and a function expansion module, in order to achieve the objective of blocking the Win32 application program interface and to change the application program used, in which the replacement function initiation module distributes storage space for the new definition function and analyzes said new definition function. The search module is used for obtaining the jump address indicated by the JMP command in the original function to be processed. The address replacement module starts from said jump address obtained by said search module in a search for all of the empty operation commands in the original function and omits these commands, and then the first function command in the original function is replaced with a JMP command, in which the address of the new definition function that is used to replace said original function is also entered in the jump address indicated by the JMP command. Finally the function expansion module reads out the contents of the registry where the new definition functions are stored and reads out sequentially any one of said new definition functions in said registry.

In the above the replacement function initiation module is used to analyze the new definition function by the following method. When said new definition function is empty, the step is then concluded and when said new definition function is not empty, a JMP command is then entered at the initial place of said new definition function as well as an assembler code to serve as a jump address of the JMP command. Said assembler code is the address of an original address to be replaced by said new definition function. Finally, everything, with the exception of the JMP command and said assembler code, in said new definition function is then emptied out and the operation is concluded in which the new definition function is analyzed by the replacement function initiation module.

In addition, in the method for obtaining said jump address in the search module, first the original function to be processed is obtained and then the current command of said original function is read out sequentially. When said command is empty then the program that processes said original function using said search module is concluded while when said command is not empty, an analysis is carried out as to whether or not said command is a JMP. When said command is not a JMP, the next command is examined by the method stated before. When said

command is a JMP, then said jump address of said command is used as a real address for said original function to be processed and the program that processes said original function using said search step is concluded.

In summary of the above, the present invention achieves its objective of modifying the application program used by directly replacing the application program indicated by the Win32 application program interface. As a result, not only is the operational speed increased but the inconvenience of having to modify the modules one by one is also eliminated with the result that omissions will not occur anymore.

In order to better understand the above and other objectives, the characteristics and advantages of the present invention, preferred application examples are given below together with attached figures to provide a more detailed explanation:

Brief description of the attached figures

Figure 1 shows a systematic block diagram of a preferred application example in accordance with the present invention.

Figure 2 shows an execution step flow chart of a preferred application example of the replacement function initiation module in accordance with the present invention.

Figure 3 shows an execution step flow chart of a preferred application example of the search module in accordance with the present invention.

Figure 4 shows an execution step flow chart of a preferred application example of the address replacement module in accordance with the present invention.

Figure 5 shows an execution step flow chart of a preferred application example of the function expansion module in accordance with the present invention.

Major part numbers

100	Systematic block diagram of the present invention
105, 110, 115	Subfunction
200	Replacement function initiation module
205-225	Execution steps of the preferred application example of the replacement function initiation module
300	Search module
305-335	Execution steps of the preferred application example of the search module
400	Address replacement module
405-430	Execution steps of the preferred application example of the address replacement module
500	Function expansion module

505-525 Execution steps of the preferred application example of the function expansion module

Preferred application example

Please refer to Figure 1. Figure 1 shows a systematic block diagram of a preferred application example in accordance with the present invention. In the systematic block diagram 100 of the present invention, a replacement function initiation module 200, a search module 300, an address replacement module 400 and a function expansion module 500 as well as subfunctions 105, 110 and 120 that are obtained by replacement by execution of these modules are included, which respectively will be described in detail in the following.

Please refer to Figure 2. Figure 2 shows an execution step flow chart of a preferred application example of the replacement function initiation module 200 in accordance with the present invention. In the replacement function initiation module 200, first distribution of the storage space is carried out for the new definition function (Step 205). Next said new definition function is checked to see whether it is an empty function (Step 210). When said new definition function is an empty function, execution of said replacement function initiation module 200 is concluded. When said new definition function is not empty, a JMP (jump) command is entered at the initial place of said new function in Step 215, and in Step 220, the address of the original function that is to be replaced by said new function is used as the assembler code, and after the JMP command is entered, said assembler code is used as the jump address indicated by said JMP command.

Next please refer to Figure 3. Figure 3 shows an execution step flow chart of a preferred application example of the search module 300 in accordance with the present invention. In the search module 300, first the original function to be processed (the standard function in the Win32 application program interface (API)) is obtained in Step 305. Next the current command of said original function (standard function) is obtained in Step 310 and then in Step 315 it is determined whether or not said command is a JMP command. When said command is a JMP command, then the operation number of said JMP command, that is the above jump address is used as a real address for said original function to be processed (Step 320). When said command is not a JMP command, the procedure moves downward to the next command (Step 325) and the operation number is moved downward by one unit (Step 330). Next it is determined whether or not said command is empty (Step 335). When said command is empty, it indicates that the end of said original function has been reached at this point and as a result the operation of said search module 300 is concluded. When said command is not empty, then the procedure returns to Step 310 to carry out a follow-up examination of said command.

Next please refer to Figure 4. Figure 4 shows an execution step flow chart of a preferred application example of the address replacement module 400 in accordance with the present invention. In the present application example, starting from the real address of the original function that is obtained by the search module 300, an analysis of said original function is first carried out by the replacement module 400 (Step 405). First it is determined whether or not the address obtained at this point belongs to said original function (Step 410). When said address is not within the range of said original function, the next address is searched for (Step 430) and the examination of said address is carried out by returning to Step 410. When said address is within the range of said original function, it is determined whether or not the command in said address is an empty operation (Step 415). When said command is an empty operation, said empty operation is omitted and Step 430 is carried out to search for the next address. When said command is not an empty operation, a JMP command is used to replace the first command 420 of the original function and the address of the new function is entered into the jump address of said JMP command (Step 425). In the above, the address obtained after the omission of the empty operation is referred to as the standard address.

Next please refer to Figure 5. Figure 5 shows an execution step flow chart of a preferred application example of the function expansion module in accordance with the present invention. In the present application example, first the contents of the registry in which the new definition function is stored are read out (Step 505). Next it is determined whether or not said registry is empty (Step 510). When said registry is completely read out, that is to say there are no follow-up data in said registry, the operation of said function expansion module is concluded. However, when said registry is not completely read out, the new definition function, that is to say the component necessary for the function expansion, is read in from the registry (Step 515). Next, the new definition function thus read out is placed at the jump address obtained by the above address replacement module (Step 520) so that the read out new definition function may be used to replace the original function to carry out an operation. Next in Step 525, the next new definition function in the registry is read out and operation is continued from Step 510 until the contents of the registry are completely read out.

As can be seen from the above the advantages of the present invention can be summarized as follows. In the present invention, not only is the operational speed increased but the inconvenience of having to modify the modules one by one is also eliminated with the result that omissions will not occur anymore.

The present invention is explained with reference to preferred examples. However, these are not to be used to limit the present invention. As is known to an ordinary person skilled in the art of the present field, various modifications and changes may be carried out without deviating

from the gist and range of the present invention. As a result the range as described in the claims should be used as the standard range of protection of the present invention.

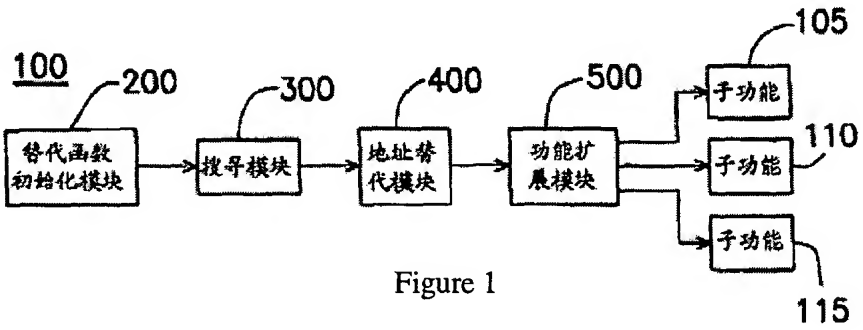


Figure 1

Key: 105, 110, 115 Subfunction
200 Replacement function initiation module
300 Search module
400 Address replacement module
500 Function expansion module

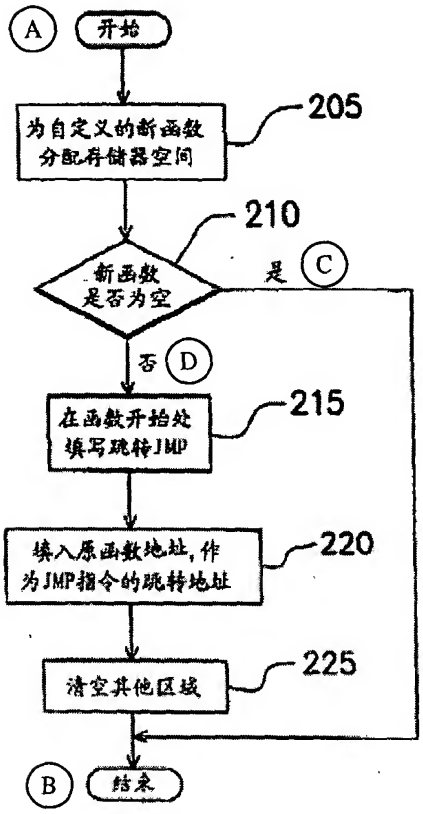


Figure 2

Key:

A	Start
B	End
C	Yes
D	No
205	Distribute storage space for the self-defined new function
210	Whether the new function is empty
215	Enter jump JMP at the initial place of the function
220	Enter the original function address to serve as the jump address of the JMP command
225	Empty out the other areas

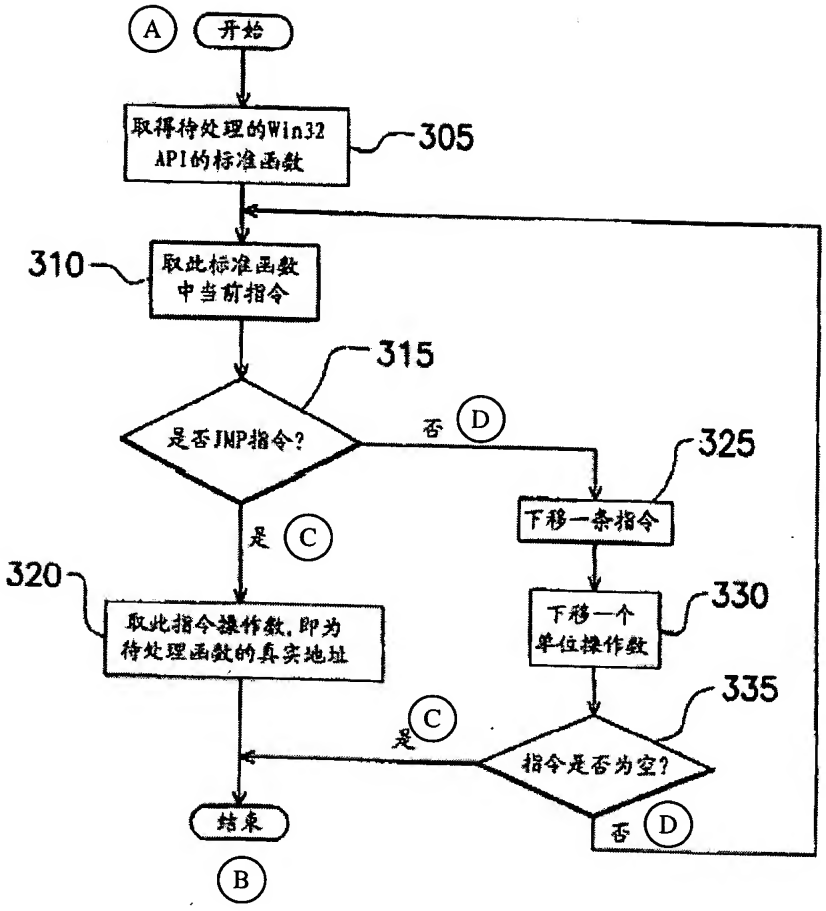


Figure 3

Key:

A	Start
B	End
C	Yes
D	No
305	Obtain the standard function of the Win32 API to be processed
310	Obtain the current command of said standard function
315	Is it a JMP command?
320	Obtain the operation number of said command, which is the read address of the function to be processed
325	Move downward to the next command
330	Move the operation number downward by one unit
335	Is the command empty?

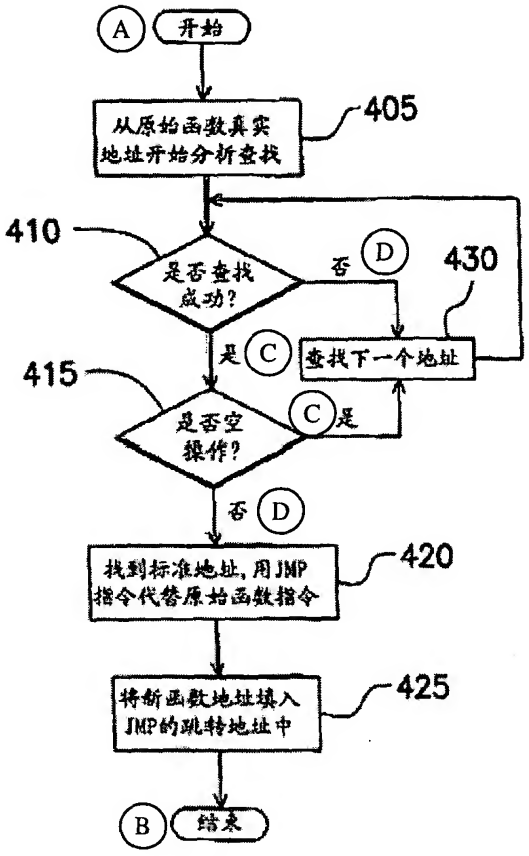


Figure 4

Key:

	A	Start
	B	End
	C	Yes
	D	No
	405	Analyze and search starting from the real address of the original function
	410	Is the search successful?
	415	Is it an empty operation?
	420	Find the standard address and replace the original function command with the JMP command
	425	Enter the new function address into the jump address of the JMP
	430	Search for the next address

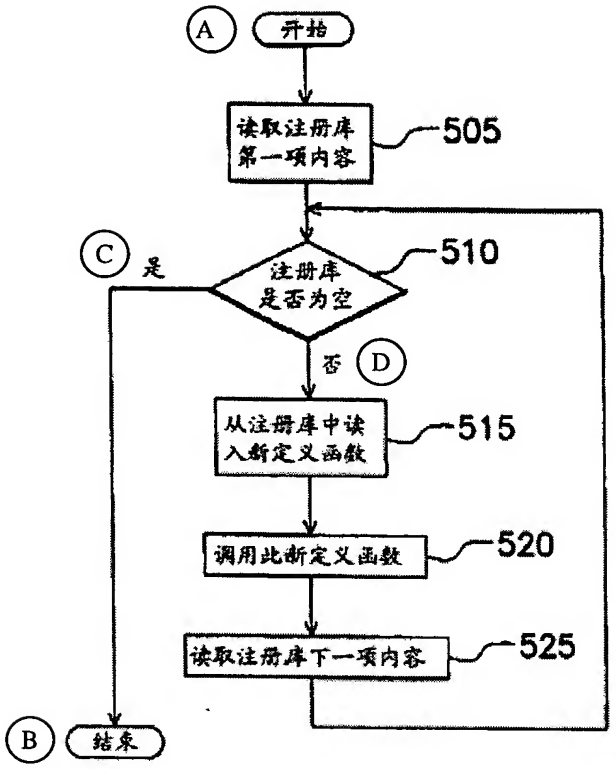


Figure 5

Key:

A	Start
B	End
C	Yes
D	No
505	Read out the first item of the contents of the registry
510	Is the registry empty?
515	Read in the new definition function for the registry
520	Adjust and use said new definition function
525	Read out the next item of the content of the registry